Unrolling Loops With Partial Hot Traces

BACKGROUND OF THE INVENTION

Field of the Invention

[0001] The present invention generally relates to computer systems, and more specifically relates to compilers that generate executable code for computer systems.

Related Patent Application

[0002] The present invention is related to, "Compiler Apparatus and Method for Unrolling a Superblock in a Computer Program", filed as serial number 10/282,811, filed on October 29, 2002, by the same inventors and owned by the current assignee at the time of the invention. The subject matter of serial number 10/282,811 is hereby included by reference in its entirety.

Description of the Related Art

[0003] Since the dawn of the computer age, computer systems have evolved into extremely sophisticated devices, and computer systems may be found in many different settings. Dramatic advances in both hardware and software (e.g., computer programs) have drastically improved the performance of computer systems. Modern software has become very complex when compared to early computer programs. Many modern computer programs have tens or hundreds of thousands of instructions. The execution time (and hence, performance) of a computer program is very closely related to the number of instructions that are executed as the computer program runs. Thus, as the size and complexity of computer programs increase, the execution time of the computer program increases as well.

[0004] Unlike early computer programs, modern computer programs are typically written in a high-level language that is easy to understand by a human programmer. Special software tools known as compilers take the human-readable form of a

computer program, known as "source code", and convert it into "machine code" or "object code" instructions that may be executed by a computer system. Because a compiler generates the stream of machine code instructions that are eventually executed on a computer system, the manner in which the compiler converts the source code to object code affects the execution time of the computer program.

[0005] The execution time of a computer program is a function of the arrangement and type of instructions within the computer program. Loops affect the execution time of a computer program. If a computer program contains many loops, or contains any loops that are executed a relatively large number of times, the time spent executing loops will significantly impact the execution time of a computer program.

[0006] In order to optimize the performance of modern computer programs, profilers have been developed to measure the run-time performance of a computer program. Profilers typically generate profile data that estimates how often different portions of the computer program are executed. Using profile data, an optimizer (such as an optimizing compiler) may make decisions to optimize loops in a computer program in order to improve the execution speed of the computer program.

[0007] Known methods for using profile data to optimize loops in a computer program do not provide an optimal solution in cases where a single hot trace (that is, a single path through which execution follows for most iterations of a loop) does not extend from a beginning of a loop to an end of the loop. As a result, the prior art may yield inefficiencies in loops that result in a slower execution time for the computer program. Application serial number 10/282,811 teaches a method for improving efficiencies in loops by identifying a hot trace and unrolling that hot trace; however additional improvements in loop efficiencies are needed to maximize performance of the computer system.

SUMMARY OF THE INVENTION

[0008] The present invention provides for loop unrolling for a class of loops that have not previously been unrolled. This class of loops comprises loops that are too large to be completely unrolled, and which lack a single hot trace that covers an entire loop iteration.

[0009] In an embodiment, a method identifies loops which contain partial hot traces, using profile data. A hot trace comprises a sequence of blocks where, with high probability, control passes from each block to the next block in the sequence. A partial hot trace is a hot trace in a loop that does not cover an entire loop iteration. The method identifies a set of instructions which constitute a proper superset of the partial hot trace and a proper subset of the entire loop and which forms a complete loop iteration. This set of instructions is then unrolled (i.e., duplicated), without unrolling the entire loop.

[0010] In an embodiment, an augmentation path set is identified. An augmentation path set has more than one path, or trace, through the augmentation path set, each trace in the augmentation path set having similar likelihood of being executed. A sum of the probabilities of executing each of the traces in the augmentation path set is similar to the probability of executing a particular block in the partial hot trace. Trace likelihood is determined using profile data.

[0011] In an embodiment, an augmentation path set lies between a beginning of the loop and a beginning of the partial hot trace. The method augments the partial hot trace by appending the partial hot trace to the augmentation path set, forming an augmented hot trace. Unrolling of the augmented hot trace is then performed as taught for unrolling a hot trace in application serial number 10/282,811.

[0012] In an embodiment, an augmentation path set lies between an end of the partial hot trace and an end of the loop. The method augments the partial hot trace by appending the augmentation path set to the end of the partial hot trace, forming an

augmented hot trace. Unrolling of the augmented hot trace is then performed as taught for unrolling a hot trace in application serial number 10/282,811.

[0013] In an embodiment, an augmentation path set lies between a first portion of the partial hot trace and a second portion of the partial hot trace. The method augments the partial hot trace by inserting the augmentation path set between the first portion of the partial hot trace and the second portion of the partial hot trace, forming an augmented hot trace. Unrolling of the augmented hot trace is then performed as taught for unrolling a hot trace in application serial number 10/282,811.

BRIEF DESCRIPTION OF THE DRAWINGS

[0014] So that the manner in which the above recited features, advantages and objects of the present invention are attained and can be understood in detail, a more particular description of the invention, briefly summarized above, may be had by reference to the embodiments thereof which are illustrated in the appended drawings.

[0015] It is to be noted, however, that the appended drawings illustrate only typical embodiments of this invention and are therefore not to be considered limiting of its scope, for the invention may admit to other equally effective embodiments.

[0016] Fig. 1A shows a group of blocks in a computer program, with each transition from one block to a subsequently executed block having profiling data shown.

[0017] Fig. 1B shows the group of blocks of Fig. 1A, but showing identification of a partial hot trace, an augmentation path set, and an augmented hot trace.

[0018] Fig. 1C shows the augmented hot trace identified in Fig. 2 unrolled.

[0019] Fig. 2A shows a group of blocks in a computer program, with each transition from one block to a subsequently executed block having profiling data shown.

[0020] Fig. 2B shows the group of blocks of Fig. 2A, but showing identification of multiple partial hot traces, the partial hot traces being separated by an augmentation path set; and an augmented hot trace.

[0021] Fig. 2C shows the augmented hot trace of Fig. 2B unrolled.

[0022] Fig. 3 is a flow diagram describing a method of unrolling loops, including unrolling augmented hot traces.

[0023] Fig. 4A shows multiple augmentation path sets and multiple partial hot traces occurring in a loop, comprising an augmented hot trace.

[0024] Fig. 4B shows the augmented hot trace of Fig. 4A unrolled.

DETAILED DESCRIPTION OF THE PREFERRED EMBODIMENTS

[0025] The present invention provides for loop unrolling for a class of loops that have not previously been unrolled. This class of loops comprises loops that are too large to be completely unrolled, and which lack a single hot trace that covers an entire loop iteration.

[0026] Turning now to Fig. 1A, a loop comprising blocks A-L is shown. Block PE is simply a loop post exit block. Profiling has been performed to determine likelihood (probabilities) of transition from one block to another, expressed as number of times the transition occurred. For example, the profiler determined that 50 transitions into the loop occurred during the period profiled. There were 250 transitions from block A to block B; 250 transitions from block A to block C. Other transition frequencies are shown next to the corresponding transition arrows. Prior compiler techniques have identified a hot trace through a loop and unrolled the trace to provide faster-running code. That is, if a sequence of blocks from a beginning of a loop to an end of a loop has a high probability of being traversed during each iteration of the loop, various trace unrolling strategies have been used. The loop shown in Fig. 1A does not have such a hot trace, since blocks B and C have similar (equal in the example of Fig. 1A)

probability of being executed. Prior trace unrolling techniques are not applicable for unrolling the loop of Fig. 1A.

[0027] Fig. 1B shows the loop of Fig. 1A with a further identification of groups of blocks. A partial hot trace 10 comprising blocks D, E, and L is identified using profiling data. Blocks D, E, and L are each executed 490 times according to the profiling data. However, Blocks D, E, and L do not make up a complete path from the beginning of the loop (i.e., block A) to the end of the loop (i.e., block L), and therefore is a partial hot trace.

[0028] A candidate augmentation path set 11 is identified using profiling data. A candidate augmentation path set comprises two or more blocks (A, B, and C in augmentation path set 11 in Fig. 1B), each block having a relatively high probability of being executed during an iteration of the loop under consideration, the candidate augmentation path set being executed in series with the partial hot trace.

[0029] A candidate augmentation path set has more than one path, or trace, through the candidate augmentation path set, each trace in the candidate augmentation path set having similar likelihood of being executed. A sum of the probabilities, or likelihoods, of executing each of the traces in the candidate augmentation path set is similar to (for example, within 25%) the probability of executing a particular block in the partial hot trace. For example, in partial hot trace 10, block D is executed 500 times per the profile data. Blocks E and L in partial hot trace 10 are executed 490 times each. In the identified candidate augmentation path set 11, block A is executed 500 times. Blocks C and D are each executed 250 times, for a total likelihood of 500. The likelihood of executing block B plus the likelihood of executing block C (total of 500) is similar to the number of executions of blocks in partial hot trace 10 (500 for block D; 490 for block E; 490 for block L).

[0030] Determination of optimal unrolling methods considers advantages of reducing loop overhead, (e.g., incrementing loop counters, testing against a limit, branching)

versus "code bloat", where repetition of code during an unrolling process introduces large amounts of executable code, inclusion of which might cause needed cache lines to be disadvantageously replaced. For example, in Fig. 1B, if block B were to represent a very complex set of instructions resulting in several thousand bytes of instructions, consuming perhaps ten or more cache lines, block B (and hence candidate augmentation path set 11) may be rejected as an augmentation path set.

[0031] Code bloat could also arise if a candidate augmentation path set comprises a large number of similarly probable traces, even if each trace is very short. For example, in an extreme case, a candidate augmentation path set could comprise 100 equally probable traces where each trace has a 1% probability of being executed. Unrolling a loop comprising such a candidate augmentation path set would result in repeating 99 unused paths in each iteration of the unroll. Such unsuitable candidate augmentation path sets are rejected as augmentation path sets. The actual number of traces having similar probabilities in a candidate augmentation path set that will be used as an augmentation path set needs to be considered based on specific characteristics of a particular computer system. In particular, cache size, cache line size, and other factors need to be considered. Advantageously, the maximum number of traces in an acceptable candidate augmentation path set is programmable, so that experimentation can be done to determine an optimum number.

[0032] Although, as described above, an augmentation path set is accepted from a candidate augmentation path set in which the number of traces having similar probability is a determinant of selection, the number of traces is typically fairly small. Advantageously in many computer systems, an augmentation path set suitable for combining with a partial hot trace comprises two or three traces of similar probability, each trace being within 25% of the probability of execution of each of the other traces. Again, actual number of traces for acceptance must be determined for a particular computer system.

[0033] Also advantageously, a selected augmentation path set comprises a relatively few instructions in each trace through the augmentation path set. Again, the number of instructions in each trace through the augmentation path set should be relatively small, and experimentation as to practical numbers of instructions in traces through the augmentation path set is necessary. The actual number, as above for the number of traces, depends on many factors relating to a particular computer system's design. Advantageously, in many computer systems, no more than ten, and preferably no more than five instructions are executed in any trace of an augmentation path set.

[0034] In Fig. 1B, augmentation path set 11 lies between the top of partial hot trace 10 and the beginning of the loop. An augmented hot trace 12 is formed by concatenation of partial hot trace 10 and augmentation path set 11. In another embodiment, an augmentation path set lies between the bottom of the partial hot trace and the end of the loop. In yet another embodiment, a first augmentation path set lies between the top of the partial hot trace and the beginning of the loop, and a second augmentation path set lies between the bottom of the partial hot trace and the end of the loop.

[0035] Fig. 1C shows an example of unrolling of augmented hot trace set 12 without unrolling the entire loop. Blocks A1, B1, C1, D1, E1, and L1 make up a first unrolled iteration of the loop; blocks A2, B2, C2, D2, E2, and L2 make up a second unrolled iteration of the loop. A1 and A2 (and similarly, B1 and B2, etc) are code instances of similarly named blocks in Figs. 1A and 1B, without the numeric suffixes (e.g., A1 and A2 are instances of A). Rarely executed code ("cold traces") is not unrolled, as doing so would tend to "bloat" the resultant code, as well as to introduce complexities that could lead to nonoptimal code, poor use of cache memory, or both. Branching to the rarely executed code (blocks F, G, H, I, J, K) is performed when needed from blocks D1 and D2. A separate instantiation of block L (L' in Fig. 1C) is created in the set of rarely executed code to complete the loop in the example, thereby avoiding a branch back into the augmented hot trace from blocks J and K. Count rectification and other

considerations of loop unrolling is performed as known by those skilled in the art, in particular, as taught in serial number 10/282,811. The number of repetitions of an augmented hot trace such as augmented hot trace 12 in Fig. 1B can be any number, as will be appreciated by those skilled in the art.

[0036] Fig. 2A shows another loop that can be considered for partial unrolling that has not been capable of being partially unrolled before. As with the example loop of Fig. 1A, exemplary profiling data is associated with each transition. The loop begins with block AX and ends with block HX. Block PEX is a post exit block. Block IX is seen to be rarely executed.

[0037] Fig. 2B shows an identification of two partial hot traces. A first partial hot trace 20A comprises blocks AX and BX. A second partial hot trace 20B comprises blocks EX, GX, and HX. Blocks CX, and DX are identified as an augmentation path set 21; profiling data showing similar frequencies of transitions through CX and DX. As discussed before, even though augmentation path set 21 has been identified as a candidate augmentation path set, it advantageously is further examined to determine if it is suitable for being repeated in an unrolling of the loop. As before, code size (e.g., number of bytes of instructions and number of traces in a particular candidate augmentation path set) must be examined for suitability for selection of the candidate augmentation path set as an augmentation path set. The example of Fig. 2B and Fig. 2C assumes selection as an augmentation path set.

[0038] Fig. 2C shows the loop of Figs. 2A and 2B partially unrolled, with two iterations of the loop partially unrolled. Similarly named blocks (with numeric suffixes) are instances of the same block of the original loop. For example, Blocks AX1 and AX2 are instances of block AX. IX is a block in a “cold trace” (seldomly executed) and is not unrolled. A separate instance HX’ of the last block in the loop (i.e., HX) is placed after IX, in order to avoid branching back into the augmented hot trace.

[0039] As before, to be accepted as an augmentation path set, the number of traces in a candidate augmentation path set and the number of instructions in a particular trace in a candidate augmentation path set are limited by considerations related to performance of the loop. These considerations include such system specifics as cache line size and cache size.

[0040] As before, the number of iterations that are unrolled is determined by tradeoffs known in the art. Any number of iterations is contemplated.

[0041] Fig. 3 shows a flowchart of an embodiment of a method to examine a program and produce partially unrolled loops, including unrolled augmented hot traces. Whereas the flowchart of Fig. 3 shows the method used once to partially unroll a loop, the method can be used for any and all instances of loops.

[0042] The method begins with step 31.

[0043] In step 32, a loop in the program's code is identified. The loop is first examined for a hot trace that extends from a beginning of the loop to an end of the loop. If such a hot trace exists, the hot trace is unrolled in step 34.

[0044] If a hot trace does not exist, control passes to step 35, where the loop is further examined for presence of a partial hot trace. If no partial hot trace is found, no unrolling of the loop is performed, as shown in block 38. If, however, one or more partial hot traces are found, control passes to step 36 which further examines the loop for existence of a candidate augmentation path set. Step 36 examines candidates for augmentation path sets for suitability, as described earlier (e.g., amount of code in each trace in the candidate augmentation path set and number of traces in the candidate augmentation path set). If one or more augmentation path sets are found, control passes to step 37, which combines the one or more partial hot traces with the one or more augmentation path sets to form an augmented hot trace. Step 39 partially unrolls the augmented hot trace with "cold blocks" not being unrolled.

[0045] Step 40 ends one iteration of the method. Typically, this method is used for each loop in the program under consideration.

[0046] Fig 4A illustrates the case where partial hot traces exist within a method, interspersed with augmentation path sets. The augmented hot trace 50 in Fig 4A is composed of a series of augmented path sets (APS1, APS2, APS3) and partial hot traces (PHT1, PHT2, PHT3). Using profile data, cold traces such as COLD in Fig 4A are excluded from the unrolling process resulting in a partially unrolled loop. PEY is the method exit point.

[0047] Fig 4B shows the loop of Fig 4A partially unrolled. In Fig 4B, each unique augmentation path set (APS) and partial hot trace (PHT) is identified by a numerical and an alphabetic suffix following the APS and PHT labels which comprise the augmented hot trace. The numerical suffix identifies each APS and PHT combination that comprises the augmented hot traces of 50A and 50B. The alphabetic suffix indicates how many instances of a particular hot trace have been partially unrolled—in the case of Fig 4B, the loop in Fig. 4A has been partially unrolled twice; once instance of augmented hot trace 50 as indicated by augmented hot trace 50A and another instance of augmented hot trace 50, augmented hot trace 50B. Using profile data, each of the unique augmented hot traces in the loop is advantageously unrolled while each cold trace, e.g. COLD in Fig 4B, is not unrolled.

[0048] The method described can be executed by a program product, such as a compiler. The program product contains instructions, that when executed by a suitable computer, perform the steps of the method. The program product resides on a computer readable media including but not limited to floppy disks, CD-ROMs, DVD disks, and magnetic tapes. The computer readable media can also be a network interface, such as the Worldwide Web, or any network coupling computers together over which program products may be transmitted. The present invention contemplates any media upon which a program product may be stored or over which a program product can be distributed.

[0049] While the foregoing is directed to embodiments of the present invention, other and further embodiments of the invention may be devised without departing from the basic scope thereof, and the scope thereof is determined by the claims that follow.